REMARKS

An Excess Claim Fee Payment Letter is submitted herewith to cover the cost of two (2) excess total claims.

Claims 1-25 are all the claims presently pending in the application. Claims 1, 2, 5-7, 11, 13-15, 17, 20 and 23 have been amended to more particularly define the invention. Claims 24-25 have been added to claim additional features of the claimed invention.

It is noted that the claim amendments are made only for more particularly pointing out the invention, and not for distinguishing the invention over the prior art, narrowing the claims or for any statutory requirements of patentability. Further, Applicant specifically states that no amendment to any claim herein should be construed as a disclaimer of any interest in or right to an equivalent of any element or feature of the amended claim.

Claims 21-22 stand rejected under 35 U.S.C. §112, first paragraph as allegedly not enabled by the specification.

Applicant notes that claims 21-22 are not subject to any prior art rejection and would, therefore, be presumably allowable except for the alleged informalities.

Claims 1-3, 5-20 and 23 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Peterson et al. (U. S. Patent No. 6,593,940) in view of Poulsen et al. (U. S. Patent No. 6,286,130). Claim 4 stands rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Peterson and Poulsen, and further in view of Flanagan et al. (U. S. Patent No. 6,343,371).

These rejections are respectfully traversed in the following discussion.

I. THE CLAIMED INVENTION

The claimed invention (e.g., as recited in claim 1) is directed to a method for statically detecting a datarace condition in a multithreaded application. The method includes inputting a set of input information, and processing the set of input information by comparing threads that may execute statements in a statement pair.

Importantly, the method includes outputting a statement conflict set that identifies the

statement pairs whose execution instances definitely or potentially cause dataraces, such that a datarace condition is statically detected without executing the multithreaded application.

Conventional datarace detection methods identify many dataraces that may never be exhibited in any program execution (e.g., "false positives"). In addition, programmer annotations are required to improve the effectiveness of such conventional tools (Application at page 6, lines 1-10).

The claimed invention, on the other hand, processes a set of input information by comparing threads that may execute statements in a statement pair, and outputs a statement conflict set that identifies the statement pairs whose execution instances definitely or potentially cause dataraces, such that a datarace condition is statically detected without executing the multithreaded application (Application at Figure 2; page 17, line 12-page 22, line 2). This feature allows the claimed method (e.g., and system) to more accurately detect a datarace condition (e.g., a condition where a datarace definitely will occur or may possibly occur) than in conventional methods (Application at page 6, lines 12-15).

II. THE 35 USC 112, SECOND PARAGRAPH REJECTION

The Examiner alleges that claims 21-22 are not enabled by the specification. Applicant submits, however, that these claims are clearly enabled by the specification.

The Examiner surprisingly asserts that "a rebuttal is yet to be received" to the unreasonable allegations which the Examiner made in the Advisory Action dated September 21, 2005. However, Applicant would point out that the statements included in the Advisory Action dated September 21, 2005 were **merely repetitive**. Indeed, **the Examiner made the same unreasonable allegations in the Office Action dated June 3, 2005**. Applicant respectfully submits that he adequately rebutted the Examiner's unreasonable allegations in the Amendment filed on July 29, 2005 and therefore, there was no reason for Applicant to respond again to the same unreasonable allegations.

Specifically, with respect to claim 21 which recites "*tagging a statement with a set of threads that may execute said statement, and comparing sets of threads for said statements*" in

the Advisory Action, the Examiner made the vague remark that "[a] lack of adequate written description issue also arises if the knowledge and level of skill in the art would not permit one skilled in the art to immediately envisage the **product** claimed from the disclosed species" (emphasis added). Presumably, by his remarks, the Examiner meant that one of ordinary skill in the art could not read the present Application and "envisage" the invention of claim 21 (which, incidentally, is directed to a method, not a product).

However, as Applicant pointed out in the Amendment filed on July 29, 2005, one of ordinary skill in the art would readily understand that the phrase "tagging a statement with a set of threads" **is a non-mathematical explanation of the following notations referred to on pages 19 and 21 in the present Application**:

MustThreadObj(p(S.i)) and MayThreadObj(p(S.i)),

where the former means the set of threads, tagged for statement p, that ***must*** execute p, and the latter means the set of threads, tagged for p, that ***may*** execute p.

With respect to claim 22 which recites "*comparing sets of locks held by threads that may execute said statements*", the Examiner stated in the Advisory Action that "synchronizing and comparing the sets of lock (sic) are not equivalent and hence, comparing the sets of locsk (sic) is still not described in Applicants (sic) specification".

However, Applicant would **AGAIN** point out that the term "lock" may refer to synchronization objects which are clearly described in the specification, for example, at page 10, line 13. **Applicant AGAIN respectfully submits that practitioners of multithreaded application developers understand well that the two terms are synonymous with each other**. Computing the synchronization objects of a statement is described in detail on page 15, lines 10 to 12 of the specification. Further, "Comparing sets of locks" is clearly described in the specification, for example, on page 19, line 24-page 20, line 8, which states:

> *"The synchronization objects of two statements allow the determination of whether two statements (or two executions of a single statement) executed by two different threads are synchronized with respect to each other via the synchronization objects (e.g., of synchronized methods and synchronized blocks)."*

Applicant would point out that the Examiner does not appear to understand the claimed invention. However, the fact that the Examiner does not understand the invention does not mean that one of ordinary skill in the art does not understand the invention.

In this case, the Examiner has provided no evidence to support his allegations.

In view of the foregoing, the Examiner is respectfully requested to withdraw this rejection.

In addition, the Examiner **surprisingly** indicated in the Office Action that claims 21 and 22 had been "withdrawn from further consideration". Applicant would point out that the Examiner's statement is EXTRAORDINARY AND HIGHLY IMPROPER. Indeed, **Applicant would point out to the Examiner that he cannot decide on a whim not to consider a pending claim. The Examiner may indicate that he disagrees with Applicant's arguments regarding claims 21 and 22. However, regardless of whether the Examiner disagrees with Applicant's arguments regarding the erroneous 35 USC 112, second paragraph rejection. In fact, the Examiner MUST examine claims 21-22,**

Therefore, Applicant respectfully requests that the Examiner retract his statement that these claims are "withdrawn from further consideration".

In view of the foregoing, the Examiner is respectfully requested to withdraw this rejection.

III. THE ALLEGED PRIOR ART REFERENCES

A. Peterson and Poulsen

The Examiner alleges that Peterson would have been combined with Poulsen to form the invention of claims 1-3, 5-20 and 23. Applicant submits, however, that these alleged references would not have been combined and even if combined, the alleged combination would not teach or suggested each and every feature of the claimed invention.

Peterson discloses a method of finding errors in multithreaded applications. Races are instanced **DURING THE EXECUTION OF THE PROGRAM.** (Peterson at Abstract).

Poulsen discloses a software implemented method for validating the correctness of

parallel computer programs. Validation is accomplished by transforming input parallel programs under computer control and **SEQUENTIALLY EXECUTING THE RESULTING TRANSFORMED PROGRAMS** (Poulsen at col. 3, lines 1-4).

Applicant respectfully submits that these references are completely unrelated, and no person of ordinary skill in the art would have considered combining these disparate references, absent impermissible hindsight.

In fact, Applicant submits that the references provide no motivation or suggestion to urge the combination as alleged by the Examiner. Indeed, these references clearly do not teach or suggest their combination. Therefore, Applicant respectfully submits that one of ordinary skill in the art would not have been so motivated to combine the references as alleged by the Examiner. Therefore, the Examiner has failed to make a prima facie case of obviousness.

However, neither Peterson, nor Poulsen, nor any alleged combination thereof teaches or suggests "*outputting a statement conflict set that identifies the statement pairs having execution instances which definitely or potentially cause dataraces, such that a datarace condition is statically detected without executing the multithreaded application*", as recited, for example, in claim 1 and similarly recited in claims 17, 20 and 23.

As noted above, unlike conventional datarace detection methods, the claimed invention outputs a statement conflict set that identifies the statement pairs whose execution instances definitely or potentially cause dataraces, such that a datarace condition is statically detected without executing the multithreaded application (Application at Figure 3; page 8, line 18-page 9, line 9; page 14, lines 9-12). This feature allows the claimed method (e.g., and system) to more accurately detect a datarace condition (e.g., a condition where a datarace definitely will occur or may possibly occur) than in conventional methods (Application at page 6, lines 12-15).

Clearly, this feature is not taught or suggested by Peterson or Poulsen. Indeed, the Examiner expressly concedes that Peterson does not teach or suggest this feature on page 5 of the Office Action, stating "*Petersen doesn't explicitly disclose performing the method statically i.e. without executing the multithreaded application*".

The Examiner surprisingly alleges that Poulsen teaches this feature of the claimed

invention. The Examiner is clearly incorrect.

In particular, the Examiner surprisingly asserts

> *"...Poulsen in an analogous art and similar configuration discloses that, 'dynamic methods that require parallel execution... are less portable, and they cannot analyze parallel programs with catastrophic errors' (Poulsen, 2:43 - 47). Therefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Petersen and Poulsen, because static detection would enable the application to analyze parallel programs more efficiently".*

From this statement is it apparent that the Examiner assumes that Poulsen had suggested a static datarace detection, **which, however, is not true**. Poulsen's invention is **NOT** a static detection of various faults in the application. Instead, his invention is simply detecting faults in a parallel application **by running the application as a sequential application, which is still a dynamic approach**. Indeed, this point is clear from the Abstract in Poulsen, which describes the Poulsen device stating:

> *A software-implemented method for validating the correctness of parallel computer programs, written in various programming languages, with respect to these programs' corresponding sequential computer programs. Validation detects errors that could cause parallel computer programs to behave incorrectly or to produce incorrect results, and is accomplished by transforming these parallel computer programs under the control of a general purpose computer and **sequentially executing the resulting transformed programs**. Validation is accomplished by detecting semantic inconsistencies between the parallel computer program and its corresponding sequential computer program. The validation method translates the input parallel*

> *computer program into a second, sequential computer program such that the second sequential computer program, **when executed**, detects and reports the semantic inconsistencies between the parallel computer program and its corresponding sequential computer program.*

Thus, it is abundantly clear that Poulsen does not teach or suggest a static datarace detection method, but a dynamic datarace detection method.

Indeed, the Examiner surprisingly attempts to rely on Poulsen's statement that "*dynamic methods that require parallel execution... are less portable, and they cannot analyze parallel programs with catastrophic errors*" (Poulsen, col. 2, lines 43 - 47). However, this passage merely refers to the fact that a parallel program's correct execution usually depends on the target (parallel) system on which the program runs while sequential program's correct execution usually does not. Thus, the Examiner's reliance on this passage is completely misguided.

The simple fact is that Poulsen's invention simply converts a parallel program into a semantically equivalent sequential program **and runs (i.e., executes) the resulting sequential program for detecting faults.** This point should be clear from the Summary of the Invention of Poulsen's shown below, particularly the third sentence:

> *"The present invention is a software-implemented method for validating the correctness of parallel computer programs that were written in various programming languages. The validation method detects errors that could cause these parallel computer programs to behave incorrectly or to produce incorrect results. Validation is accomplished by transforming input parallel computer programs under computer control **and sequentially executing the resulting transformed programs**. The validation method is system-independent, is portable across various computer architectures and platforms, and requires no input program modification on the part of the programmer".*

The examiner also claims that *"[t]herefore it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Petersen and Poulsen, because static detection would enable the application to analyze parallel programs more efficiently."* However, even assuming (arguendo) that people have suggested applying static analysis to datarace detection does not make it obvious to one of ordinary skill in the art how such a static analysis could be performed.

Indeed, **dynamic analysis and static analysis are fundamentally different**, and knowing how to perform a dynamic analysis to solve or detect a problem does not necessarily mean knowing how to perform a static one to do the same. Dynamic analysis uses the information on what has actually happened during an execution while a static analysis is to infer, based on the program text, what may happen during an execution. For example, dynamically detecting a null-pointer error or a divide-by-zero error is trivial because runtime systems or hardware usually detect them and report.

However, combining that they are trivial with somebody's suggesting a static detection for them does not make how to do static detection of any of them trivial or obvious to anyone. **Although some of them can be easily detected statically, they are in general very difficult to detect statically.**

Therefore, Applicant submits that these alleged references would not have been combined and even if combined, the combination would not teach or suggest each and every feature of the claimed invention. Therefore, the Examiner is respectfully requested to withdraw this rejection.

B. Flanagan

The Examiner alleges that the Peterson would have been combined with Poulsen, and further combined with Flanagan to form the invention of claim 4. Applicant submits, however, that these alleged references would not have been combined and even if combined, the alleged

combination would not teach or suggested each and every element of the claimed invention.

Flanagan discloses a system for statically detecting potential race conditions in a multi-threaded computer program. In the system, a race condition detector determines whether accesses to object data fields are consistently protected by an appropriate lock. An object data field access that is not protected by an appropriate lock indicates a potential race condition (Flanagan at Abstract).

Applicant respectfully submits that these references would not have been combined as alleged by the Examiner. Indeed, these references are completely unrelated, and no person of ordinary skill in the art would have considered combining these disparate references, absent impermissible hindsight.

In fact, Applicant submits that these references do not include any motivation or suggestion to urge the combination as alleged by the Examiner. Indeed, these references clearly do not teach or suggest their combination.

Therefore, Applicant respectfully submits that one of ordinary skill in the art would not have been so motivated to combine the references as alleged by the Examiner. Therefore, the Examiner has failed to make a prima facie case of obviousness.

Moreover, Applicant submits that neither Peterson, nor Poulsen, nor Flanagan, nor any alleged combination thereof, teaches or suggests "*outputting a statement conflict set that identifies the statement pairs having execution instances which definitely or potentially cause dataraces, such that a datarace condition is statically detected without executing the multithreaded application*", as recited, for example, in claim 1 and similarly recited in claims 17, 20 and 23.

Clearly, this feature is not taught or suggested by Flanagan. Indeed, Flanagan simply states the properties that need be checked, which are the conditions for a datarace. What matters is **how the properties are checked**, which is an important aspect of the claimed invention.

However, that is at least one area where Flanagan's invention differs from the claimed invention. Using the analogy of the null-pointer error or the divide-by-zero error above,

Flanagan's statement corresponds to saying that null-pointers and divide-by-zero need be checked, which is, needless to say, true for null-pointer or divide-by-zero checking.

In short, nowhere does Flanagan teach or suggest outputting a statement conflict set that identifies the statement pairs whose execution instances definitely or potentially cause dataraces, such that a datarace condition is statically detected without executing the multithreaded application. Thus, Flanagan is clearly unrelated to the claimed invention.

Thus, Flanagan clearly does not make up for the deficiencies of Peterson and Poulsen.

Therefore, Applicant submits that these alleged references would not have been combined and even if combined, the combination would not teach or suggest each and every element of the claimed invention. Therefore, the Examiner is respectfully requested to withdraw this rejection.

III. FORMAL MATTERS AND CONCLUSION

Applicant respectfully requests that the Examiner confirm his receipt and acceptance of the formal drawings filed herein on March 26, 2002.

Applicant notes that claim 2 has been amended to address the Examiner's objections thereto. Applicant further notes that the terms "*IsPotentialDR relation*" and "*IsDefiniteDR relation*" are clearly defined in the Application at pages 20-21, and therefore, it is unlikely that one of ordinary skill in the art would read the Application and not understand the meaning of these two terms.

In view of the foregoing, Applicant submits that claims 1-26, all the claims presently pending in the application, are patentably distinct over the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary in a telephonic or personal interview.

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully Submitted,

Date: 4/17/06

Phillip E. Miller, Esq.
Registration No. 46,060

McGinn IP Law Group, PLLC
8321 Old Courthouse Road, Suite 200
Vienna, VA 22182-3817
(703) 761-4100
Customer No. 21254

CERTIFICATE OF FACSIMILE TRANSMISSION

I hereby certify that the foregoing Amendment was filed by facsimile with the United States Patent and Trademark Office, Examiner Chuck D. Kendall, Group Art Unit # 2192 at fax number (571) 273-8300 this 17th day of April, 2006.

Phillip E. Miller
Reg. No. 46,060